

66/60/11



JC674 U.S. PTO

# UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications  
under 37 CFR 1.53(b))

Attorney Docket No. **0100.9900960** Total Pages 30  
First Inventor Don A. Van Dyke  
Title Multi-Instruction Set Flag Preservation  
Apparatus and Method  
Express Mail Label No. EL286432505US

JC690 U.S. PTO  
09/436882



11/09/99

<b>APPLICATION ELEMENTS</b> See MPEP chapter 600 concerning utility patent application contents.	<b>ADDRESS TO:</b> Assistant Commissioner for Patents Box Patent Application Washington, DC 20231
---	---

1. ☒ **Fee Transmittal Form**  
(Submit an original, and a duplicate for fee processing)
2. ☒ **Specification** Total Pages 17  
(preferred arrangement set forth below)
  - Descriptive title of the Invention
  - Cross References to Related Applications
  - Statement Regarding Fed sponsored R & D
  - Reference to Microfiche Appendix
  - Background of the Invention
  - Brief Summary of the Invention
  - Brief Description of the Drawings (if filed)
  - Detailed Description
  - Claim(s)
  - Abstract of the Disclosure

3. ☒ **Drawings (35 USC 113) Total Sheets 5**
4. **Oath or Declaration** Total Pages 2
  - a. ☒ Newly executed (original or copy)
  - b. ☐ Copy from a prior application  
(37 CFR 1.63(d))  
(for continuation/divisional with Box 17 completed)  
[Note Box 5 below]

- i. ☐ **DELETION OF INVENTOR(S)**  
Signed statement attached deleting  
inventor(s) named in the prior application,  
see 37 CFR 1.63(d)(2) and 1.33(b).

5. ☐ Microfiche Computer Program (Appendix)
6. ☐ Nucleotide and/or Amino Acid Sequence  
Submission (if applicable, all necessary)
  - a. ☐ Computer Readable Copy
  - b. ☐ Paper Copy (identical to computer copy)
  - c. ☐ Statement verifying identity of above  
copies

## ACCOMPANYING APPLICATION PARTS

7. ☒ Assignment Papers (cover sheet & document(s))
8. ☒ 37 CFR 3.73(b) Statement ☒ Power of  
(when there is an assignee) Attorney
9. ☐ English Translation Document (if applicable)
10. ☐ Information Disclosure ☐ Copies of  
Statement (IDS)/PTO-1449 IDS Citations
11. ☐ Preliminary Amendment
12. ☒ Return Receipt Postcard (MPEP 503)  
(Should be specifically itemized)
13. ☐ Small Entity ☐ Statement filed in Prior  
Statement(s) Application, Status still  
proper and desired.
14. ☐ Certified Copy of Priority Document(s)  
(if foreign priority is claimed)
15. ☐ Other

16. If a **CONTINUING APPLICATION**, check appropriate box and supply the requisite information:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No:  
Prior Application Information: Examiner Group / Art Unit:

## 17. CORRESPONDENCE ADDRESS

☐ Customer Number or Bar Code Label

or, ☒ Correspondence Address Below

Markison & Reckamp, P.C.  
175 West Jackson Boulevard - Suite 1015  
Chicago, Illinois 60604  
Telephone: 312-939-9800 Facsimile: 312-939-9828

Name (Print/Type)	Christopher J. Reckamp	REGISTRATION NUMBER	34,414
Signature		Date	Nov. 9, 1999

**PATENT APPLICATION**  
**DOCKET NO. 0100.9900960**

**In the United States Patent and Trademark Office**

**FILING OF A UNITED STATES PATENT APPLICATION**

**Title:**

**MULTI-INSTRUCTION SET FLAG PRESERVATION  
APPARATUS AND METHOD**

**Inventors:**

<b>Name: Don A. Van Dyke 5133 Independence Drive Pleasanton, California</b>	
---	--

**Attorney of Record**  
**Christopher J. Reckamp**  
**Registration No. 34,414**  
**175 W. Jackson Blvd. – Suite 1015**  
**Chicago, Illinois 60604**  
**Phone (312) 939-9800**  
**Fax (312) 939-9828**

Express Mail Label No. EL286432505US

Date of Deposit: November 9, 1999

I hereby certify that this paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addresses" service under 37 C.F.R. Section 1.10 on the 'Date of Deposit', indicated above, and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Name of Depositor: Rosalie Swanson  
(print or type)

Signature: Rosalie Swanson

## MULTI-INSTRUCTION SET FLAG PRESERVATION APPARATUS AND METHOD

### Field Of The Invention

The invention relates generally to multi-instruction set processing devices and methods and more particularly to multi-instruction set processing devices and methods incorporating instruction emulation of instructions from one instruction set using instructions from the other instruction set.

### Background Of The Invention

Microprocessors and other instruction execution devices are known that employ variable length instruction sets, such as Intel® X86 family of microprocessors. Also, processors are known that execute different instruction sets, such as a variable length instruction set (e.g., X86 type instructions) and other instruction sets such as fixed length RISC instruction sets. With multi-instruction set processors, variable length instructions are sometimes converted to a plurality of fixed length native instructions to speed up execution of the variable length instruction. For example, an X86 based processor, may use a plurality of RISC instructions that may be fixed length instructions, to represent one or more variable length instructions. Typically, the RISC instructions are executed out of an onboard memory and are user accessible. An arithmetic logic unit, for example, may then receive the RISC instructions and execute the RISC instructions at a more efficient rate. Accordingly, integer instructions and other instructions may be converted from a non-native instruction to one or more native instructions.

However, a problem can arise when variable length instructions, such as non-native instructions, require the setting of flags in various flag registers for particular

instructions. For example, if a set of variable length instruction are emulated using a plurality of fixed length native instructions and the fixed length native instructions update the flag registers that are used by other non-native instructions, the updating of the flags in the various flag registers during emulation can corrupt the state used by instructions being executed for other arithmetic logic units or other processor elements relying on the other variable length instructions being executed that also rely on the flag settings. One method of solving such a problem may be to save and restore flag states, but this can result in excessive overhead requirements.

Consequently, there exists a need for a method and apparatus for processing program instructions in a multi-instruction set processing device, that helps suitably control the preservation of flag settings for variable length instructions that are emulated using fixed length native instructions.

### **Brief Description Of The Drawings**

The invention will be more readily understood in view of the following drawings wherein:

Figure 1 is a block diagram illustrating one example of a multiple processing device apparatus that employs use of native instructions having flag modification enable data in accordance with one embodiment of the invention;

FIG. 2 is a block diagram illustrating one example of an apparatus for processing program instructions in accordance with one embodiment of the invention;

FIG. 3 is a diagram illustrating one example of a native instruction format that includes flag modification enable bits in accordance with one embodiment of the invention;

FIG. 4 is a graphic illustration depicting native emulation for nonconvertible variable length instructions and flag modification for converted variable length instructions in accordance with one embodiment of the invention; and

FIG. 5a and FIG. 5b is a flow chart illustrating one example of the operation of the apparatus shown in FIG. 2.

## Detailed Description Of a Preferred Embodiment of The Invention

Briefly, a method and apparatus for processing program instructions, utilizes native fixed length instructions that include at least one flag modification enable bit. The flag modification enable bit is typically sent with the operation code and other information in the native instruction and is set to allow updating or prevention of updating of one or more flags, such as flags stored in flag registers associated with non-native instructions, such as variable length instructions. As such, a flag modification enable bit(s) may be set to preserve flag bit setting for variable length instructions that are emulated using the fixed length native instructions, to prevent overwriting of flag settings during emulation of variable length instructions.

FIG. 1 illustrates an example of a multi-instruction set processing device 10 that includes a plurality of processing units 12a and 12b, such as microprocessors. The processing device 10 may be, for example, an integrated circuit. The processing device 10 is operatively coupled to other processing devices (not shown) through a suitable bus 14, such as a PCI bus used in personal computers, hand held devices and other portable devices. Memory 16, such as dynamic RAM (DRAM) or any other suitable memory, stores a software instruction emulation module 18 that is used to emulate variable length instructions that are not converted into a plurality of fixed length native instructions, as set forth below.

If desired, the processing device 10 may include a memory gateway 20 and an I/O gateway 22. The memory gateway 20 may be a conventional memory gateway that is operatively coupled to the plurality of processing units 12a and 12b through a general bus 24. Similarly, the I/O gateway 22 may serve as a gateway to allow communication with I/O chips through PCI bus 14. The I/O gateway 22 is operatively coupled to the plurality of processing units 12a and 12b through the bus 24. Each of the processors 12a and 12b receives native instructions 26 or non-native instructions over the general bus 24 through a suitable communication bus 28a and 28b. The native instruction 26 may be, for example, a fixed length instruction from a fixed length instruction set having operational code 30 and one or more flag modification enable bits 32, along with other suitable

instruction information. The non-native instruction may be variable byte length instructions, such as X86 type instructions.

The software instruction emulation module 18 contains programming instructions to allow the emulation of variable length instructions, using a plurality of native instructions 26. The software instruction emulation module 18 is preferably run as needed by the plurality of processors 12a and 12b. In a preferred embodiment, the software instruction emulation module 18 is requested to emulate non-native instructions, such as complex variable length instructions, using native instructions 26 containing the flag modification bit 32. The non-native instructions that are emulated are preferably those instructions that are not convertible by an instruction set converter. For example, the emulated variable length instructions may be very complex X86 instructions that may be more quickly implemented and verified using the software emulation module 18 as opposed to a hardware based execution scheme. Some examples of X86 instructions that may be emulated include far call, task gate call and CPUID instructions. The software instruction emulation module 18 has the option of preventing the modification of flags, such as flags set when performing integer operations or other mathematical operations, in a flag register in response to execution of the plurality of native instructions used by the software instruction emulation module 18. Some emulated instruction modify the flags, while others preserve the flag settings. The software emulation module 18 contains instructions with the flag modification bit 32 set to inform the controller to modify or not modify flag settings as required.

In contrast, when variable length instructions are suitably converted into fixed length native instructions by a converter, the arithmetic logic units executing the instructions are allowed to update the associated flags in the flag registers for those converted variable length instructions.

Each of the processors 12a and 12b receive native instructions containing data representing operational code and data representing at least one flag modification enable bit 32 as part of the native instruction. The processors 12a and 12b determine whether

the flag modification enable bit 32 allows updating of ALU related flags, in response to executing the operational code. Each of the processors 12a and 12b may update the ALU flags in response to determining the status of the flag modification enable bit 32. For example, if the flag modification enable bit 32 is set to a logical high, indicating that flag modification is enabled, a processor may modify flags in a flag register in response to executing the native code during ALU execution of the native instructions. In contrast, if the flag modification enable bit is set, for example, to a logic zero, indicating flag modification is prevented, the processors are unable to write to the distributed flag register (FIG. 2), as is typically done during emulation of more complex variable length instructions.

FIG. 2 illustrates one example of an apparatus 200 for processing program instructions that may be included in each of the processing units 12a and 12b. In this example, the apparatus 200 includes an instruction cache 202, an instruction multiplexer 204, an instruction cache controller 206, a first buffer 208, an instruction aligner 210, an instruction aligner controller 212, an instruction converter 214, a multiplexer 216, a second buffer 218, and a plurality of arithmetic logic units 220a-220n that may include control logic 222. One or more flag registers 224 contain flags that are modified as required by the variable length instructions being executed. The flag registers may be distributed flag registers, meaning flag registers that are located throughout the processor, as desired.

The bus 24 may transfer variable length instructions, referred to herein as non-native instructions, and fixed length instructions, referred to herein as native instructions. Native instructions are typically communicated to processing units 12a and 12b through bus 24 from memory 16 and cached in instruction cache 202. Variable length instructions (e.g., non-native instructions) are typically communicated through a bus 24 from memory 16.

The instruction cache 202 may be any suitable instruction cache that can cache variable length instructions, as well as fixed length instructions. The multiplexer 204

receives instructions 226 from the instruction cache 202 and/or instructions directly from the bus 24, indicated as instructions 228. The instruction cache controller 206, as known in the art, controls the multiplexer 204 to output selected instructions 230 for execution by, for example, one or more ALUs or other instruction execution stages. The selected instructions 230 are buffered in the first buffer 208 to allow the instruction aligner 210 to suitably align the variable length instructions under control of aligner controller 212.

Any suitable variable length instruction alignment mechanism may be used. For example, a suitable instruction alignment mechanism may be one such as that found in co-pending application entitled "Variable Length Instruction Alignment Device and Method," filed by T.R. Ramesh et al., having attorney docket no. 0100.990098, owned by instant Assignee and incorporated herein by reference. Once the selected variable length instructions are suitably aligned, the converter 214 converts the one or more aligned variable length instructions to a plurality of native fixed length instructions through the use of hardwired logic indicated as 240. The aligned variable length instructions 238 may also be of a complex type such that conversion is not appropriate.

Converted instructions typically result in fewer than eight native instructions. When converting, the converter 214 converts, for example, variable length X86 instructions to a plurality of resulting native instructions 240. The plurality of resulting native instructions 240 may include at least one flag modification enable bit 32 set to allow changing of non-native instruction flags in the flag register 224, in response to execution of the plurality of native instructions, by, for example, an ALU instruction execution stage.

However, for the unconvertible non-native instructions 242, the converter generates an unconvertible instruction command indicating that emulation of the non-native instruction should be performed since conversion was not appropriate. The converter 214 generates the unconvertible instruction command in response to detecting that the X86 instruction is not convertible by the converter 214. The controller 222, upon receiving the unconvertible non-native instruction then requests the software instruction emulation module 18 to emulate the unconvertible non-native instruction 242. If a native



instruction is received on the bus 24, no conversion is necessary and the native instruction 244 will pass to the multiplexer 216.

The controller 222 may be any suitable control logic and may be a part of one or more ALUs 220a-220n, or part of a converter or any other suitable block. The controller 222 controls the multiplexer 216 to output native instruction 244 and native instructions 240 resulting from a conversion, or unconvertible non-native instruction 242 into the second buffer 218. The controller 222 knows which of the input information 240 and 244 to output to the second buffer, based on which instruction is being executed.

Accordingly, the controller 222 outputs an address select signal 250 to control the multiplexer 216 to output a selected instruction 252 for storage in the second buffer 218. The second buffer 218, receives the instruction containing the operational code and data representing at least one flag modification enable bit in the form of native instruction 244 or a native instruction 240 resulting from conversion. The controller 222 emits a

command to buffer 218 to supply ALUs 220a-220n with instructions. The controller 222 analyzes the instruction 256 taken from the second buffer 218 to determine whether the flag modification enable bit 32 (assuming a native instruction) allows modification of a flag in the flag register, in response to executing the operational code embedded in the native instruction. The controller 222 accordingly receives flag status data 258 indicating the status of the flag modification enable bit 32. If the flag modification enable bit 32 indicates allowance of flag updates for the flag register, the controller 222 generates a flag update command 260 to update one or more flags in the flag register 224 in response to determining the status of the one flag modification enable bit 32. The flag register 224 is operatively coupled to the controller 222 so that the controller can update a flag in the flag register if the native instruction flag modification native bit 32 is set to allow modification of the flag in the flag register. The ALUs perform integer instructions, e.g., add, subtract, etc., in native instructions then the control logic (controller) checks if the flag modification enable bit is set and updates flag reg if the bit is not set. The controller 222 may also, if desired, provide the ALU execution stage with pipeline control information 262 indicating, for example, source operand availability.

In this embodiment, where the ALUs execute the native instructions containing the flag modification enable bits, the received instructions 256 include at least one of an integer instruction, an arithmetic instruction and a logical instruction. However, it will be recognized that non-integer instructions may also employ flag modification enable bits as desired. The ALUs may be any suitable arithmetic logic units, and may be, for example, of a type described in co-pending application entitled "Method and Apparatus of Configurable Processing," filed by inventors Korbin Van Dyke et al., having attorney docket no. 0100.9900930, filed on or about August 18, 1999 owned by instant assignee and incorporated herein by reference.

Referring to FIG. 3, one example of a native instruction 26 contains the flag modification enable bit 32, first operational code 30 (op code), size information 300, indicating, for example, the size of the operation which may be, for example, a 16-bit, 32-bit, 64-bit, 128-bit or any other suitable size, destination register of the operation 302 indicating, for example, where to store results of the operation, first source register information 304 indicating the first register containing input data to the operation, second op code 306, second source information indicating the second register containing input data, and any other suitable information as desired. The native instruction 26 may be, for example, an add, subtract, multiply, divide, increment, decrement, negation, rotate, shift, XOR, AND, OR, or any other suitable instruction, for example, executable by the arithmetic logic units. The flag modification enable bit 32 may be designed so that if the bit is a "0" the flag settings in the flag registers cannot be affected through execution of the native instruction containing the flag modification bit, or if the bit is set to a logic 1, this state may indicate that the flag value may be modified based on the operation performed consistent with the native instruction.

The native instruction 26 as shown includes at least one input operand and at least one destination operand (302 and 304), respectively.

The software instruction emulation module 18, when executed by one or more of the processing units 12a, serves as a variable length instruction emulator that uses fixed

length native instructions to emulate variable length instructions. Programmers of software emulation module 18 typically set the flag modification enable bit of each native instruction to preserve flag bit settings for variable length instructions that are emulated using the fixed length native instructions. The non-native instruction emulator 18

5 emulates unconverted variable length X86 instructions, for example, using a plurality of native instructions wherein the native instructions include the flag modification enable bit that may be set to prevent changing of non-native instruction flags in response to execution of the plurality of native instructions that are used to emulate the unconverted variable length instruction.

10

FIG. 4 graphically illustrates a queue of X86 instructions 400 wherein X86 instruction 402 is not convertible. It is therefore emulated using native instructions from the native instruction set 404. In addition, the converter produces a sequence of native instructions. Where the native instructions have been generated based on conversion, the

15 flag modification operation 406 is performed to modify one or more of the necessary flags in the flag register 224 as required by the native instructions. Flag register 224, by way of example, and not limitation, indicates various flags that may be set. For example, a zero flag, overflow flag, parity flag, sign flag, carry flag, or any other suitable flags.

20

FIGS. 5a and 5b illustrate one example of the operation of the system of FIG. 2 wherein the process includes storing received non-native variable length instructions in the instruction cache as shown in block 500. Since the non-native instructions are variable length and fetched in fixed length groups, the process includes aligning the variable length instructions, as shown in block 502, using the aligner 210. As shown in

25 block 504, the converter, or other suitable logic, determines if the received variable length instruction is directly convertible to a plurality of native instructions. If the variable length instruction is unconvertible, the process continues as shown in block 506, by designating the instruction as unconverted, by using, for example, an unconvertible instruction command. This unconvertible instruction is then emulated by the software

30 instruction emulation module. This may be carried out, for example, by the CPU calling the emulation module upon detection of the unconvertible instruction command. The

unconvertible instruction command may be any suitable data, such as any bits or any other suitable information such as appended to or indexed with the unconvertible variable instruction.

- 5           However, if the aligned variable length instruction is convertible to a plurality of native instructions containing the flag modification enable bit, the process continues as shown in block 508 to convert the non-native instruction to native instructions. For example, an X86 ADD instruction where one operand is memory and the sequence is
- LDA T, MEM ADD
- 10   ADD Dest, T

- As shown in block 512, the buffers or ALUs receive the native instructions containing the flag modification enable bits. The appropriate ALU then executes the resulting native instructions from the conversion as shown in block 514. As shown in
- 15   block 516, the ALU determines whether the flag modification enable bit allows updating of the flag in a non-native instruction flag register, such as register 224. As shown in block 518, since the native instructions have been generated through conversion, the flag modification enable bit should be set so that it allows setting of flags. Accordingly, the process includes updating one or more flags in the flag register in response to
- 20   determining the status of the modification enable bit. The process then ends for that instruction and continues for other instructions, as shown in block 520.

- Referring back to block 506, if the instruction received from the bus 24 has been designated as an unconverted instruction so that it is passed for emulation by the
- 25   instruction emulator, the process includes, as shown in block 522, emulating the unconvertible non-native instructions using native instructions containing the flag modification enable bits. The native instructions used to emulate the unconvertible non-native instructions typically have the flag modification enable bits set to prevent updating of the flag register 224 to avoid overwriting of flags that may be being set by other
- 30   instructions that are being executed. As shown in block 524, the process includes determining whether the flag modification enable bit allows updating of the flag in the

non-native instruction flag register (flag register 224) so that, for example, if the bit is set to prevent changing, as shown in block 526, the process includes preventing changing of flags for the emulated instructions based on the state of the flag of the modification enable bit for a given native instruction.

5

Accordingly, the above apparatus and methods can provide multi-instruction set processing units with faster execution by allowing some instructions to be converted for direct execution by an ALU. However, for those instructions that require emulation, flags may be prevented from being modified during the execution of emulation to  
10 eliminate the need to save and restore the flag registers. Hence, the disclosed apparatus and method allow flexibility in the ordering of instructions and the modification of flags and use of flags.

It should be understood that the implementation of other variations and  
15 modifications of the invention in its various aspects will be apparent to those of ordinary skill in the art, and that the invention is not limited by the specific embodiments described. For example, the apparatus and methods may be implemented using any suitable combination of hardware, software and firmware. Also, the functions of one element may be shared or varied to other suitable elements as desired. It is therefore  
20 contemplated to cover by the present invention, any and all modifications, variations, or equivalents that fall within the spirit and scope of the basic underlying principles disclosed and claimed herein.

## Claims

WHAT IS CLAIMED IS:

1. A method for processing program instructions comprising the steps of:  
5 receiving at least one instruction containing data representing operational code and data representing at least one flag modification enable bit;  
determining whether the at least one flag modification enable bit allows  
updating of at least one flag in response to executing the operational code; and  
updating at least one flag in response to determining a status of the at least  
10 one flag modification enable bit.
2. The method of claim 1 wherein the at least one instruction is at least one of: an integer instruction, an arithmetic instruction and a logical instruction.
- 15 3. The method of claim 1 wherein the at least one instruction includes at least one input operand.
4. The method of claim 1 wherein the step of updating at least one flag in response to determining a status of the at least one flag modification enable bit, includes  
20 updating a flag register if the flag modification enable bit is set to allow modification of a flag in the flag register.
5. The method of claim 1 including the steps of:  
providing a variable length instruction emulator that uses fixed  
25 length native instructions as the at least one instruction, to emulate variable length instructions; and  
evaluating the flag modification enable bit to preserve flag bit settings for variable length instructions that are emulated using the fixed  
length native instructions.

30

6. The method of claim 1 wherein the at least one instruction is a fixed length instruction.
7. The method of claim 1 including the step of emulating non-native instructions using native instructions containing the flag modification bit.
8. The method of claim 7 wherein the non-native instructions are variable length X86 instructions.
9. The method of claim 8 including the steps of:
  - converting variable length X86 instructions to a plurality of native instructions wherein the plurality of native instructions include the at least one flag modification enable bit set to allow changing of non-native instruction flags in response to execution of the plurality of native instructions; and
  - emulating unconverted variable length X86 instructions using a plurality of native instructions wherein the native instructions include the at least one flag modification enable bit set to prevent changing of non-native instruction flags in response to execution of the plurality of native instructions.

10. An apparatus for processing program instructions comprising:
  - a buffer coupled to receive at least one instruction containing data representing operational code and data representing at least one flag modification enable bit; and
  - a controller operatively responsive to the at least one instruction stored in the buffer, that determines whether the at least one flag modification enable bit allows updating of at least one flag in response to executing the operational code and that updates at least one flag in response to determining a status of the at least one flag modification enable bit.
11. The apparatus of claim 10 wherein the at least one instruction is at least one of: an integer instruction, an arithmetic instruction and a logical instruction.
12. The apparatus of claim 11 wherein the at least one instruction includes at least one input operand.
13. The apparatus of claim 10 including a flag register operatively coupled to the controller, wherein the controller updates a flag in the flag register if the flag modification enable bit is set to allow modification of the flag in the flag register.
14. The apparatus of claim 10 including a variable length instruction emulator that uses fixed length native instructions as the at least one instruction, to emulate variable length instructions.
15. The apparatus of claim 10 wherein the at least one instruction is a fixed length instruction.
16. The apparatus of claim 14 wherein the variable length instruction emulator emulates non-native instructions using native instructions containing the flag modification bit.



17. The apparatus of claim 16 wherein the non-native instructions are variable length X86 instructions.

18. The apparatus of claim 17 including:

5           an instruction converter, operatively coupled to receive variable length X86 instructions, that converts the received variable length X86 instructions to a plurality of native instructions wherein the plurality of native instructions include the at least one flag modification enable bit set to allow changing of non-native instruction flags in response to execution of the plurality of native instructions;

10       and

          a non-native instruction emulator operatively responsive to an unconvertible instruction command generated in response to detecting that an X86 instruction is not convertible by the converter, wherein the non-native instruction emulator emulates unconverted variable length X86 instructions using a plurality of native instructions wherein the native instructions include the at least one flag modification enable bit set to prevent changing of non-native instruction flags in response to execution of the plurality of native instructions for the unconverted variable length instruction.

15

20

# MULTI-INSTRUCTION SET FLAG PRESERVATION APPARATUS AND METHOD

## Abstract Of The Invention

5

A method and apparatus for processing program instructions, utilizes native fixed length instructions that include at least one flag modification enable bit. The flag modification enable bit is typically sent with the operation code and other information in the native instruction and is set to allow updating of one or more flags, such as stored in flag registers, associated with non-native instructions, such as variable length instructions. In addition, a flag modification enable bit may be set to preserve flag bit setting for variable length instructions that are emulated using the fixed length native instructions, to prevent overwriting of flag settings during emulation of variable length instructions.

10

11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2

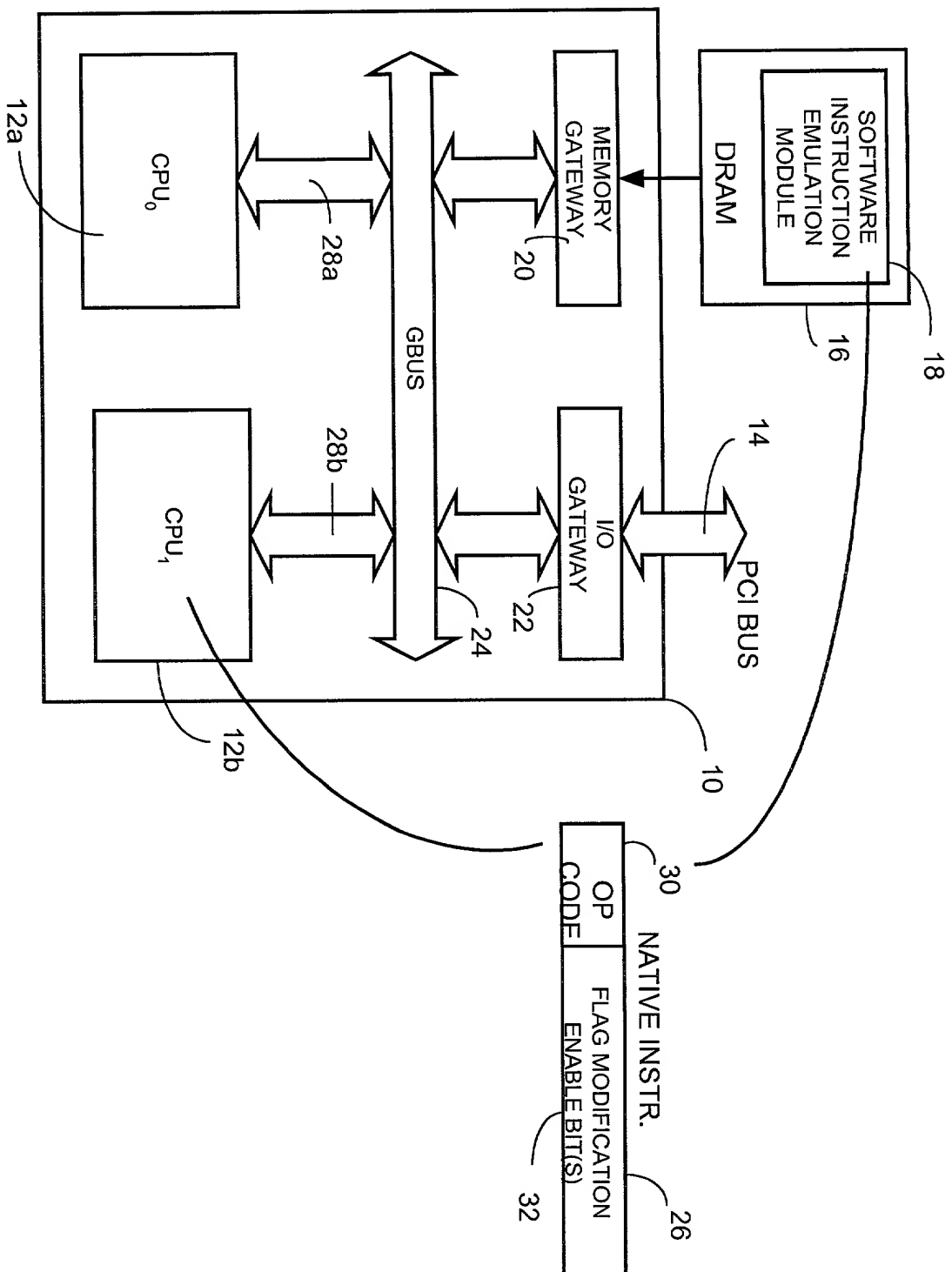


FIG. 1

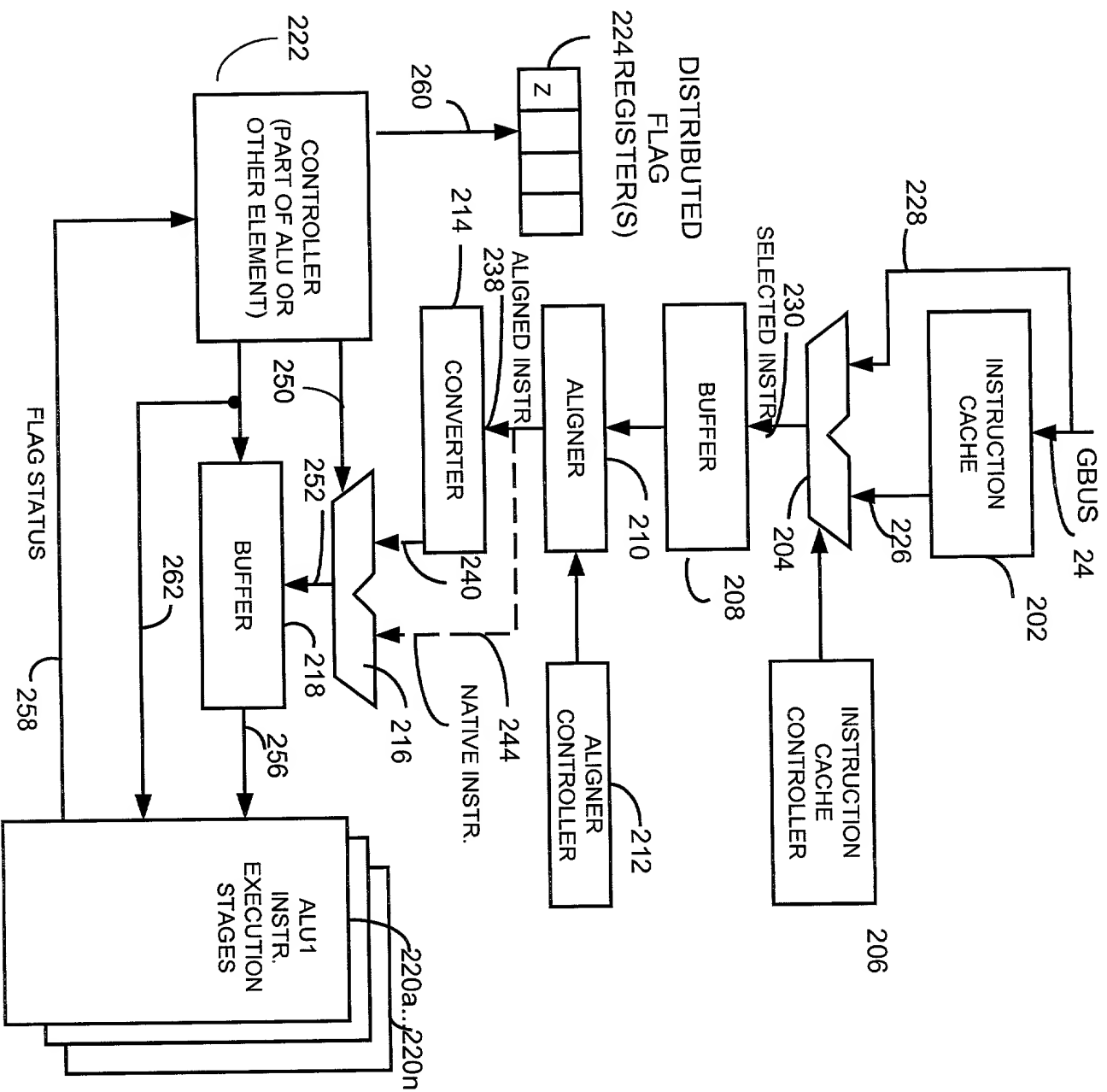


FIG. 2

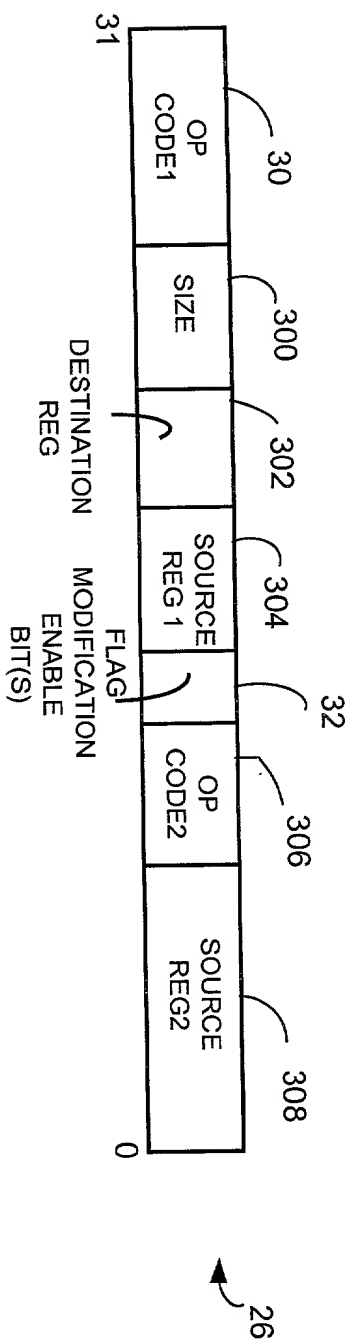


FIG. 3

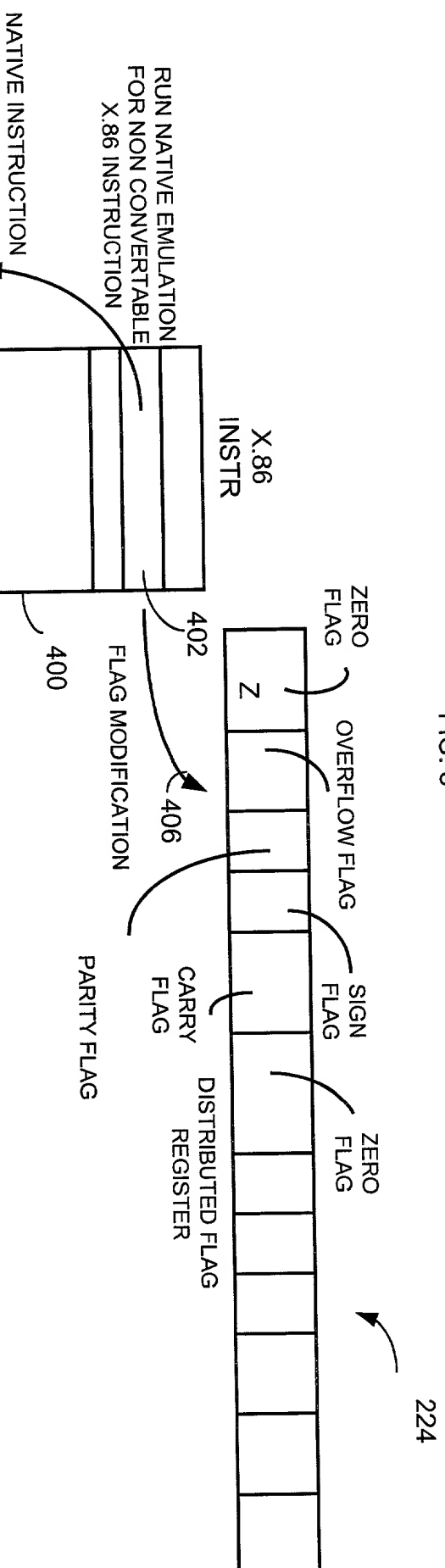


FIG. 4

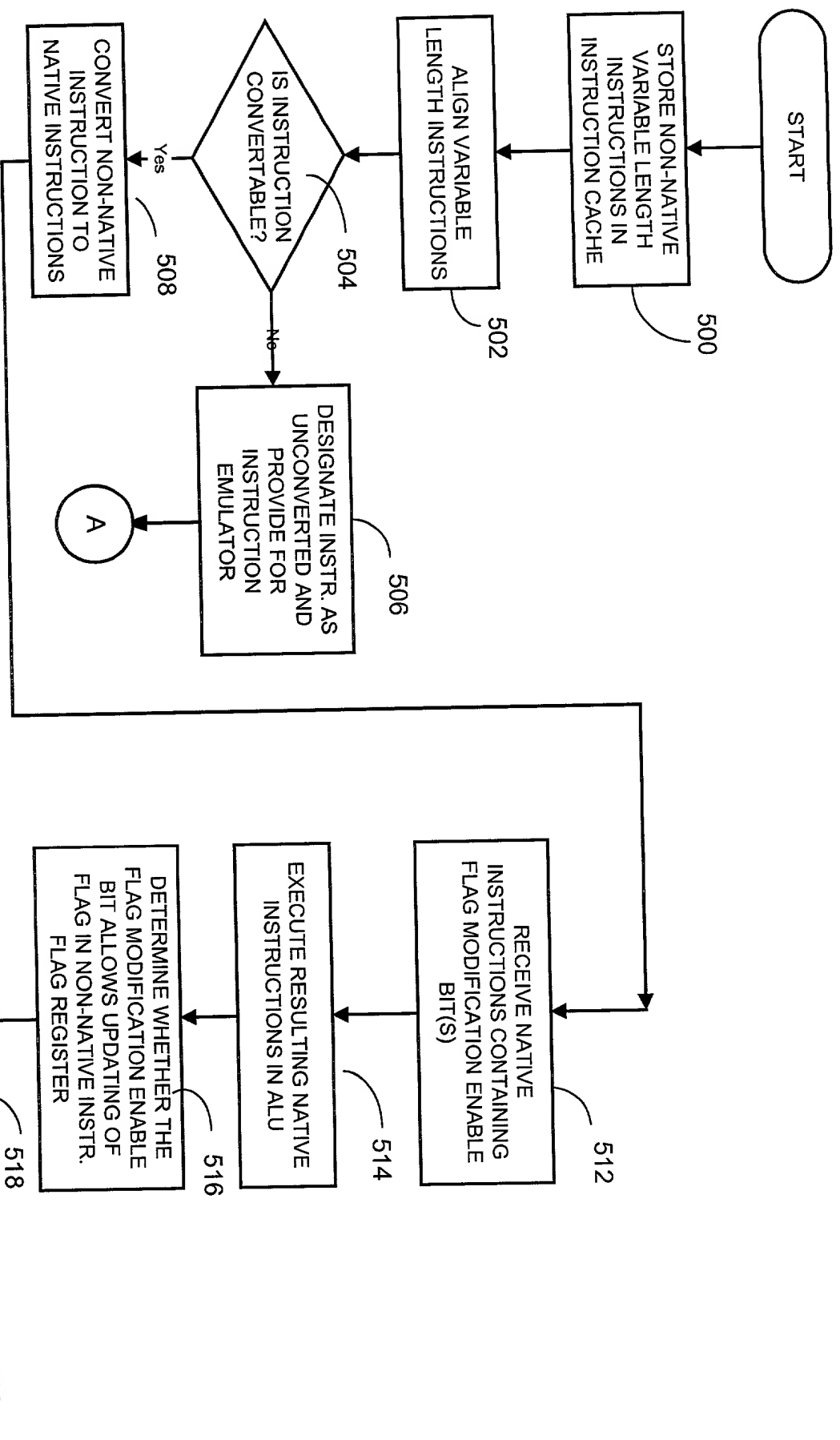


FIG. 5a

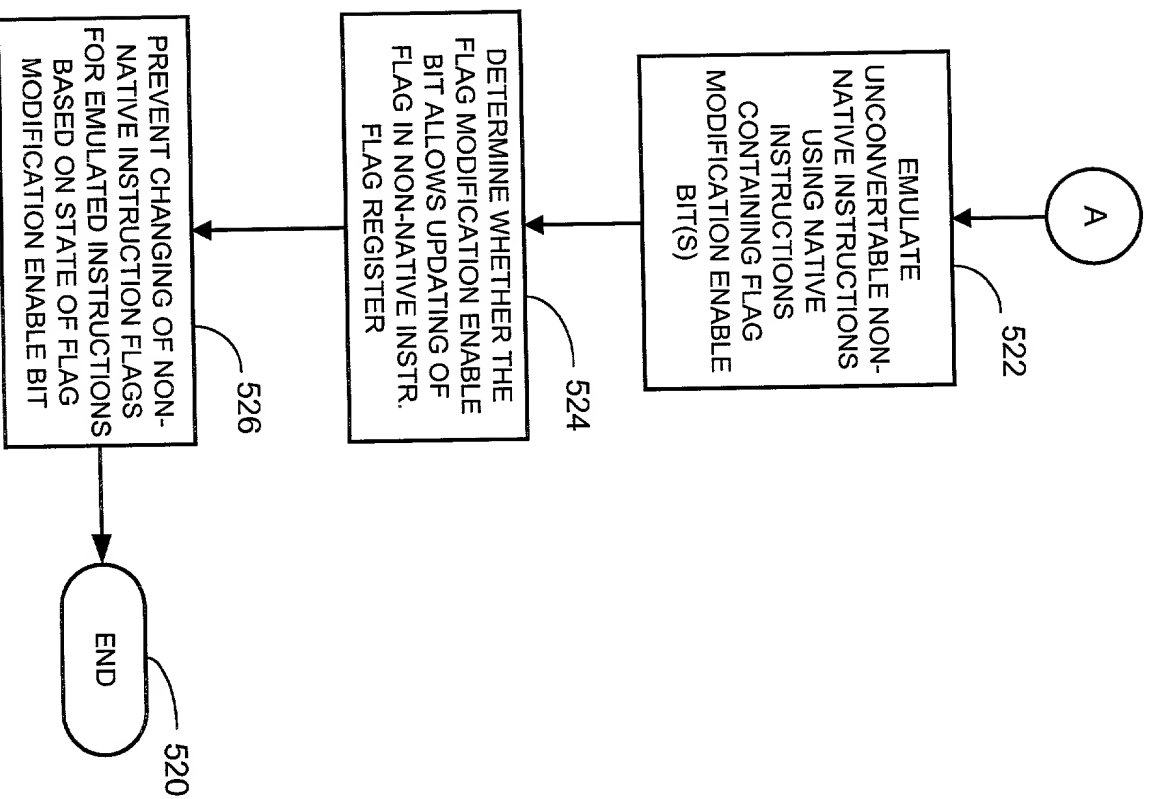


FIG. 5b

# DECLARATION FOR UTILITY OR DESIGN PATENT APPLICATION

(37 CFR 1.63)

- ☒ Declaration Submitted with Initial Filing, OR  
☐ Declaration Submitted after Initial Filing  
(surcharge (37 CFR 1.16 (e)) required)

Attorney Docket Number 0100.9900960

First Named Inventor Don A. Van Dyke

COMPLETE IF KNOWN

Application Number

Filing Date

Group Art Unit

Examiner Name

As a below named inventor, I hereby declare that:

My residence, post office address, and citizenship are as stated below next to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: **MULTI-INSTRUCTION SET FLAG PRESERVATION APPARATUS AND METHOD**, the specification of which:

☒ is attached hereto.

☐ was file on (MM/DD/YYYY) as United States Application Number or PCT International Application Number and was amended on (MM/DD/YYYY) (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment specifically referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56.

I hereby claim foreign priority benefits under 35 U.S.C. 119(a)-(d) or 365(b) of any foreign application(s) for patent or inventor's certificate, or 365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or of any PCT international application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application Number(s)	Country	Foreign Filing Date (MM/DD/YYYY)	Priority Not Claimed	Certified Copy Attached?	
				YES	NO
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

☐ Additional foreign application numbers are listed on a supplemental priority data sheet PTO/SB/02B attached hereto.

I hereby claim the benefit under 35 U.S.C. 119(e) of any United States provisional application(s) listed below.

Application Number(s)	Filing Date (MM/DD/YYYY)

☐ Additional provisional application numbers are listed on a supplemental priority data sheet PTO/SB/02B attached hereto.

I hereby claim the benefit under 35 U.S.C. 120 of any United States application(s), or 365(c) of any PCT international application designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 U.S.C. 112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application.

U.S. Parent Application or PCT Parent Number	Parent Filing Date (MM/DD/YYYY)	Parent Patent Number (if applicable)

☐ Additional U.S. or PCT international application numbers are listed on a supplemental priority data sheet PTO/SB/02B attached hereto.



As a named inventor, I hereby appoint the following registered practitioner(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Name	Registration Number	Name	Registration Number
Timothy W. Markison	33,534	Christopher J. Reckamp	34,414
Paul M. Anderson	39,896		
Sally Daub	41,478		

☐ Additional registered practitioner(s) named on supplemental Registered Practitioner Information sheet PTO/SB/02C attached hereto.

Direct all correspondence to:

**Markison & Reckamp, P.C.**  
**175 West Jackson Boulevard - Suite 1015**  
**Chicago, Illinois 60604**  
**Telephone: 312-939-9800**  
**Facsimile: 312-939-9828**

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

**Name of Sole or First Inventor:**

☐ A petition has been filed for this unsigned inventor

Given Name (first and middle [if any])		Family Name or Surname	
Don A...		Van Dyke	
Inventor's Signature	<i>Don A. Van Dyke</i>	Date	11/4/99
Residence	City: Pleasanton	State: CA	Country: USA
Post Office Address		5133 Independence Drive	
City: Pleasanton	State: CA	ZIP: 94566	Country: USA

**Name of Additional Joint Inventor:**

☐ A petition has been filed for this unsigned inventor

Given Name (first and middle [if any])		Family Name or Surname	
Inventor's Signature		Date	
Residence	City:	State:	Country:
Post Office Address		Citizenship:	
City:	State:	ZIP:	Country:

**Name of Additional Joint Inventor:**

☐ A petition has been filed for this unsigned inventor

Given Name (first and middle [if any])		Family Name or Surname	
Inventor's Signature		Date	
Residence	City:	State:	Country:
Post Office Address		Citizenship:	
City:	State:	ZIP:	Country:

☒ Additional inventors are being named on the 1 supplemental Additional Inventor(s) sheet PTO/SB/02A attached hereto.